NONLINEAR OPTIMIZATION

Qingsha Cheng 程庆沙

Multiobjective Optimization

Formulation of the multiobjective optimization problem

Pareto dominance and optimal sets

Totally conflicting, nonconflicting and partially conflicting objectives

Solution techniques for multiobjective optimization

Introduction to multiobjective evolutionary algorithms

Elitism, archiving and density assessment

What is Multiobjective Optimization?

a problem of finding **a vector of decision variables** which **satisfies constraints** and **optimizes a vector function** whose elements represent the objective functions.

These functions form a mathematical description **of performance criteria** which are usually **in conflict with each other**.

Hence, the term “optimize” means finding such a solution which would give the **values of all the objective functions acceptable to the designer**.

Examples of Optimization Problems Involving Multiple Objectives

1. Bridge construction: minimizing **total mass** and maximizing **stiffness** at the same time
2. Aircraft design: optimization of **fuel efficiency**, **payload** and **weight**
3. Chemical plant design: **total investment** and **net operating cost**
4. Sunroof design in a car: minimizing **noise** and maximizing the **ventilation**
5. Portfolio optimization problem: minimizing **risk** and maximizing the **fiscal return**
6. Electrical engineering: improving performance of the device (e.g., maximizing **gain** and **bandwidth**, minimizing **noise** and **nonlinear distortion**) and minimizing **size** and **power** consumption at the same time

Formulation of Multiobjective Optimization Problem

Solve

$$x^* \in \arg \min_{x \in X \subseteq R^n} F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix}$$

where x is an n-dimensional vector of decision variables, and X is a feasible set

$F(x)$ is a objective vector with m objective components to be minimized, which may be competing or non-commensurable to each other

Formulation of Multiobjective Optimization Problem

Formulation with explicit constraints: solve

$$x^* \in \arg\min_{x \in R^n} F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix}$$

subject to

$$c_{eq.k}(x) = 0 \qquad k = 1, 2, ..., n_{eq} \qquad \text{(equality constraints)}$$

and

$$c_{ineq.k}(x) \leq 0 \qquad k = 1, 2, ..., n_{ineq} \qquad \text{(inequality constraints)}$$

Vilfredo Pareto

(born Wilfried Fritz Pareto; Italian: [vilˈfreːdo paˈreːto]; 15 July 1848 – 19 August 1923) was an Italian engineer, sociologist, economist, political scientist, and philosopher, now also known for the 80/20 rule, named after him as the Pareto principle.

Pareto Dominance and Optimal Sets

Named after Vilfredo Pareto

The scalar concept of “optimality” does not apply directly in the multiobjective setting

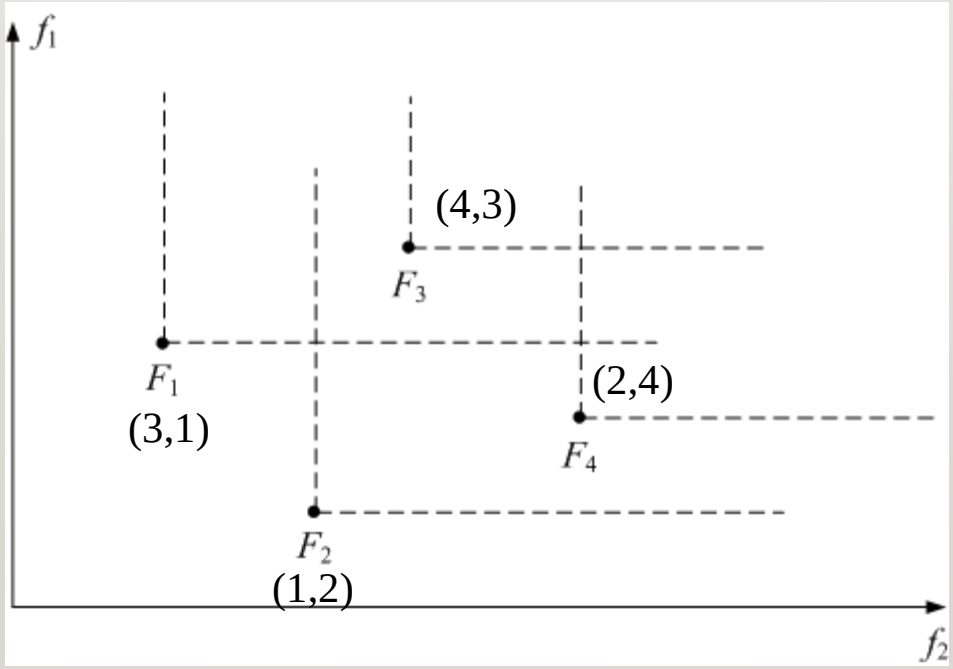
Instead, the concept of *Pareto dominance* is used:

Definition: An objective vector F_a in a minimization problem is said to dominate another objective vector F_b, denoted as $F_a \preceq F_b$ if and only if $f_{a.i} \leq f_{b.i}$ for all $i = 1,\ldots,m$, and there is $j \hat{I} \{1,\ldots,m\}$ such that $f_{a.j} < f_{b.j}$.

Remark: without loss of generality, we shall consider minimization problems; maximization of any given objective function f_i can be turned into minimization of $-f_i$

Pareto Dominance and Optimal Sets

Illustration of Pareto dominance:

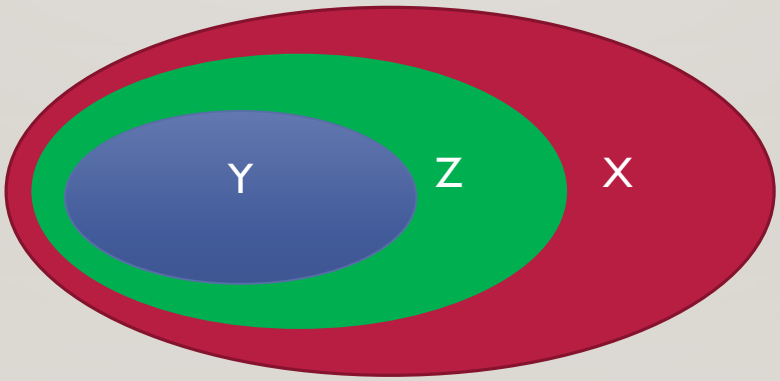


F_1 is not dominated by neither of F_2, F_3 nor F_4
F_2 is not dominated by neither of F_1, F_3 nor F_4
F_3 is dominated by both F_1 and F_2 but not by F_4
F_4 is dominated by F_2 but not by F_1 nor F_3

Pareto Dominance and Optimal Sets

Definition: A subset Y of the feasible set X is called a *local Pareto optimal set* with respect to a subset Z, $Y \subseteq Z \subseteq X$, if there is no $z \in Z$ such that $F(z) \preceq F(y)$ for some $y \in Y$.

Definition: A subset Y of the feasible set X is called a *global Pareto optimal set* if there is no $x \in X$ such that $F(x) \preceq F(y)$ for some $y \in Y$.



Pareto Dominance and Optimal Sets

Definition: A subset Y of the feasible set X is called a *local Pareto optimal set* with respect to a subset Z, $Y \subseteq Z \subseteq X$, if there is no $z \in Z$ such that $F(z) \preceq F(y)$ for some $y \in Y$.

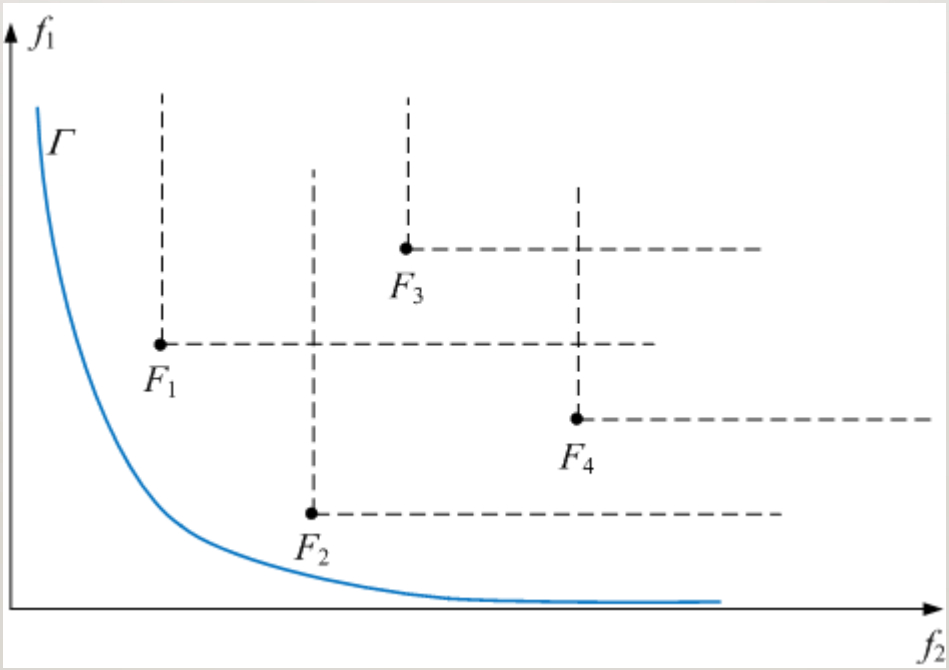
Definition: A subset Y of the feasible set X is called a *global Pareto optimal set* if there is no $x \in X$ such that $F(x) \preceq F(y)$ for some $y \in Y$.

Remark: Local Pareto optimal set would refer to a Pareto optimal set found at each iteration or at the end of the optimization in a single run. Global Pareto optimal set refers to the actual Pareto optimal set for multiobjective optimization.

Remark: We will refer to objective vectors corresponding to locally (globally) optimal Pareto sets as locally (globally) non-dominated.

Pareto Dominance and Optimal Sets

Illustration of Pareto optimal sets:



The set $Y = \{F_1, F_2\}$ is locally non-dominated in $Z = \{F_1, F_2, F_3, F_4\}$

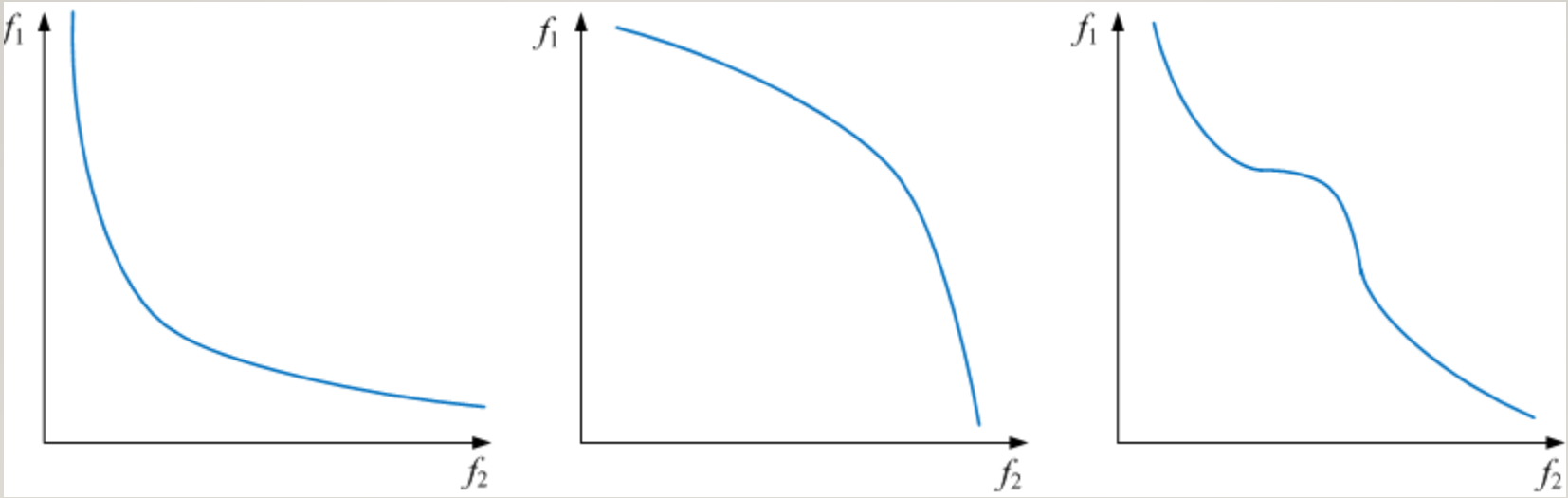
The set G is globally non-dominated

Pareto Dominance and Optimal Sets

Globally non-dominated set of objective vectors is also called *Pareto Front*.

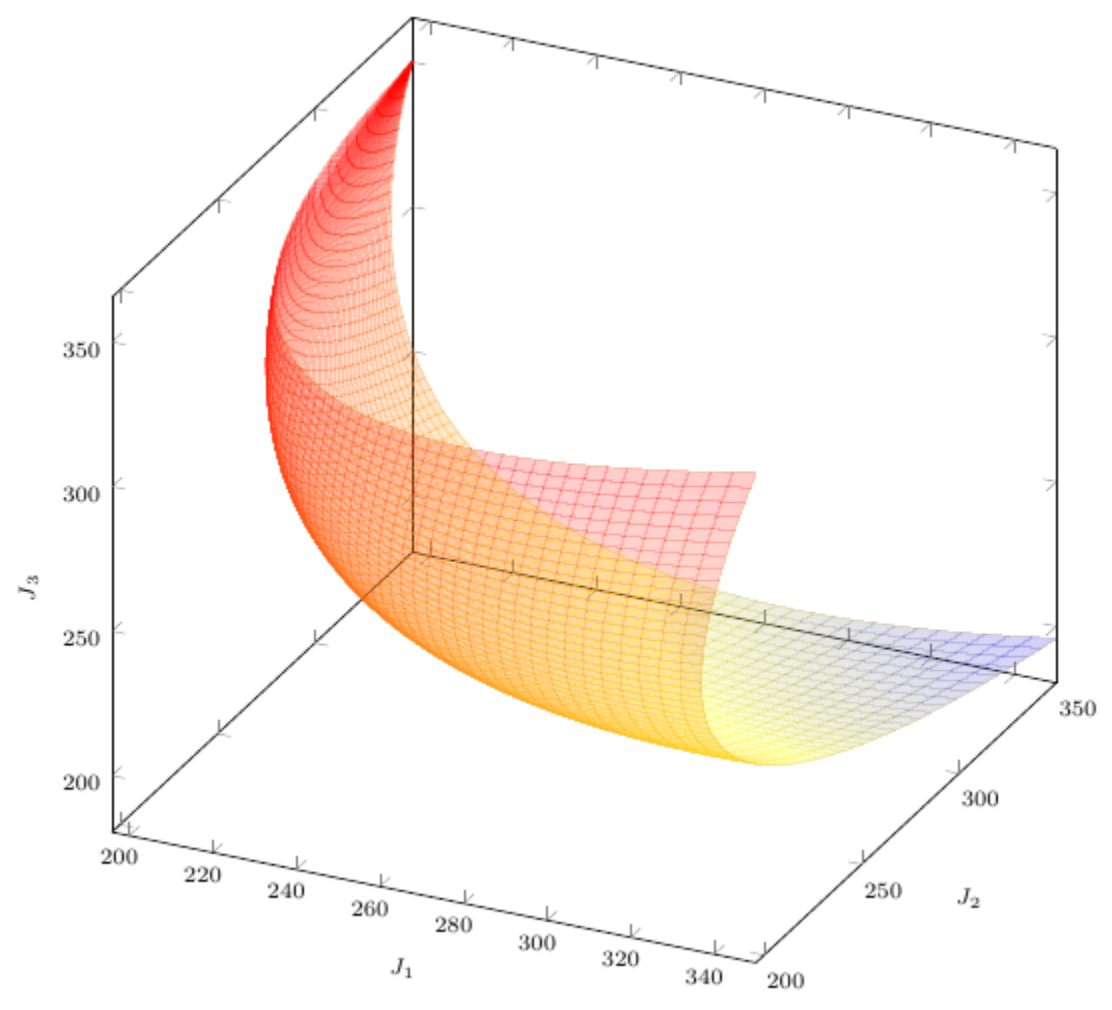
Pareto front for an *m*-objective optimization problem is at most an *m*-1 surface

Examples of Pareto fronts for $m = 2$ (f_1 has only one corresponding f_2)



Pareto Dominance and Optimal Sets

Examples of Pareto fronts for $m = 3$

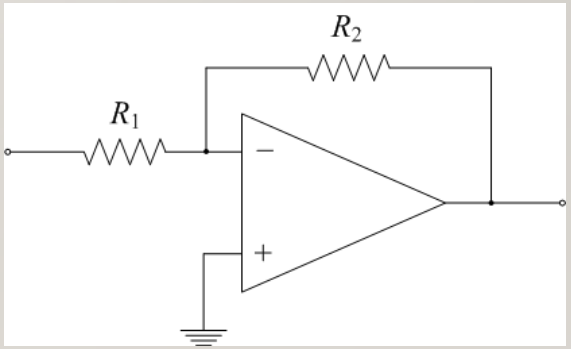


Totally Conflicting Objective Functions

Definition: A vector $F = [f_1 \ldots f_m]^T$ of objective functions is said to be *totally conflicting* if there exists **no two solutions** $x, y \hat{I} X$ such that $F(x) \preceq F(y)$ or $F(y) \preceq F(x)$.

Remark: No optimization is necessary for this class of multiobjective problems because **the feasible set** X already represents **the global Pareto optimal solutions.**

Example: Find R_1 and R_2 of the inverting amplifier so that both its gain A and bandwidth B are maximized.



For this circuit we have $A \approx -R_2/R_1$ and $A \cdot B \approx$ const.
=> maximizing gain and bandwidth are totally conflicting objectives

Nonconflicting Objective Functions

Definition: A vector $F = [f_1 \ldots f_m]^T$ of objective functions is said to be *nonconflicting* if for any **two solutions** $x, y \hat{\mathrm{I}} \ X$ we have either $F(x) \preceq F(y)$ or $F(y) \preceq F(x)$.

Remark: This class of multiobjective problems can be converted into single-objective problems, either by arbitrarily considering only one of the objective components during the optimization or by combining the multiple objectives into a scalar function; any improvement for one objective component will lead to the improvement of the remaining components, and vice versa.

Example: Given the stiffness of the steel bridge, minimize its weight and the cost of material used to construct it.
Here, both objectives are nonconflicting as minimizing the weight will reduce the amount of material and its cost at the same time.

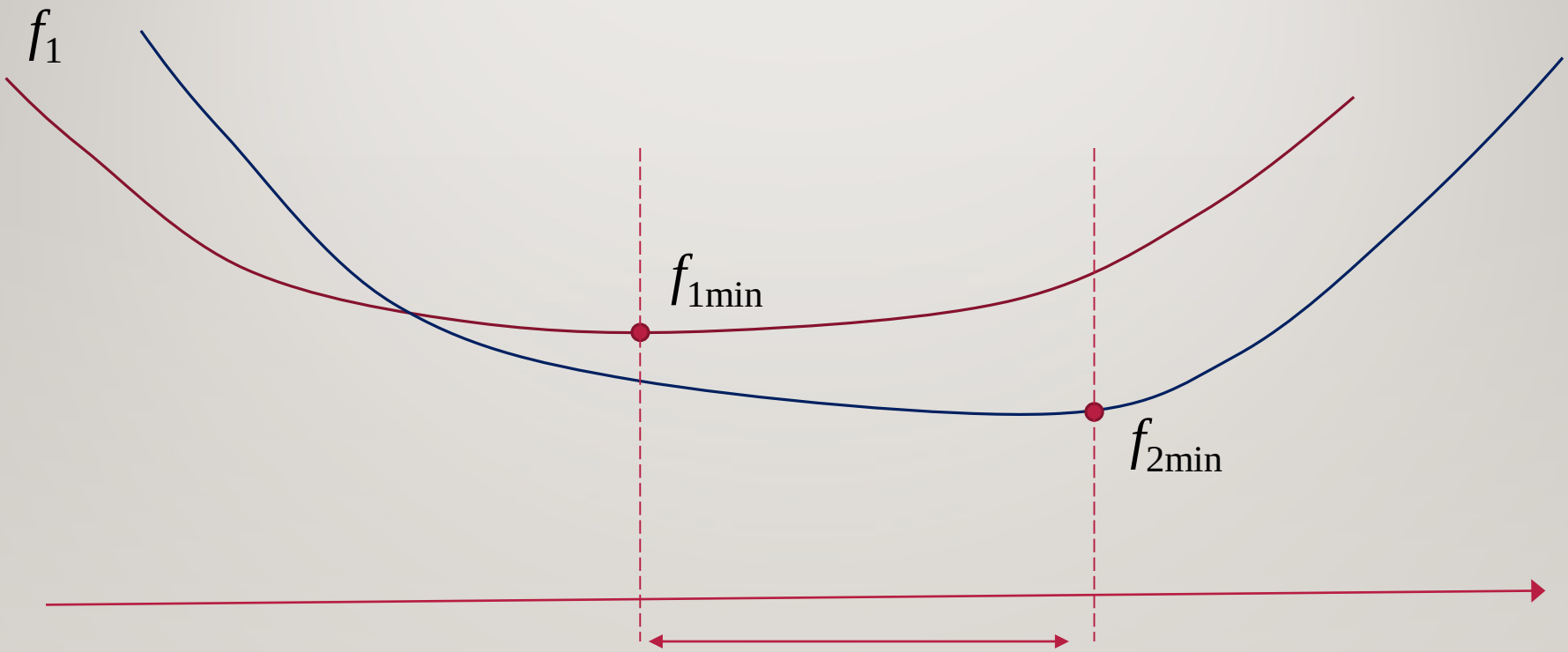
Partially Conflicting Objective Functions

Definition: A vector $F = [f_1 \ldots f_m]^T$ of objective functions is ***partially conflicting*** if it is neither totally conflicting nor nonconflicting.

Remark: Many real-world design optimization tasks belong to this class of multiobjective optimization problems. Here, a set of Pareto optimal solutions representing the trade-offs among the conflicting objectives are desired.

Example: The circuit consists of functional units u_i, $i = 1, \ldots, N$, connected according to connectivity matrix $E = [e_{ij}]$. The area and power dissipation of unit u_i is A_i and P_i, respectively. Divide the circuit into K blocks so that the number of interconnections is minimized while the area and power dissipation is as equally distributed between the blocks as possible.

Partially Conflicting Objective Functions



Solution Techniques for Multiobjective Optimization

In multiobjective optimization, a specific and compromised decision is often needed based upon the set of trade-off solutions.

The final solution for the optimization results from both the process of **optimization** and **decision making (choice)**, which can be one of the following:

1. **Priori preference articulation**: transform multiobjective problem into a single-objective problem prior to the optimization.
2. **Progressive preference articulation**: decision and optimization are intertwined where partial preference information is provided upon which optimization occurs.
3. **Posteriori preference articulation**: a set of efficient candidate solutions is found from a certain method and then a decision is made on the choice of the best solution

Solution Techniques for Multiobjective Optimization

If the relative importance of different objectives is clearly defined and known beforehand, **priori preference articulation** seems to be a proper approach.

If the knowledge of the problem is limited, initial goals may be set for objectives and then modified during the optimization process, especially if it turns out that some objectives are difficult to be met (**progressive preference articulation**).

In general, whenever possible, it is preferable to have as much information on the Pareto optimal set as we can get, which would give an idea about possible trade-offs between design objectives and allow us to select the most suitable solution (**posteriori preference articulation**). This approach might be **very expensive**.

Solution Techniques for Multiobjective Optimization

Weighted sum method:

A standard technique for multiobjective optimization is to minimize a positively weighted convex sum of the objectives, i.e,

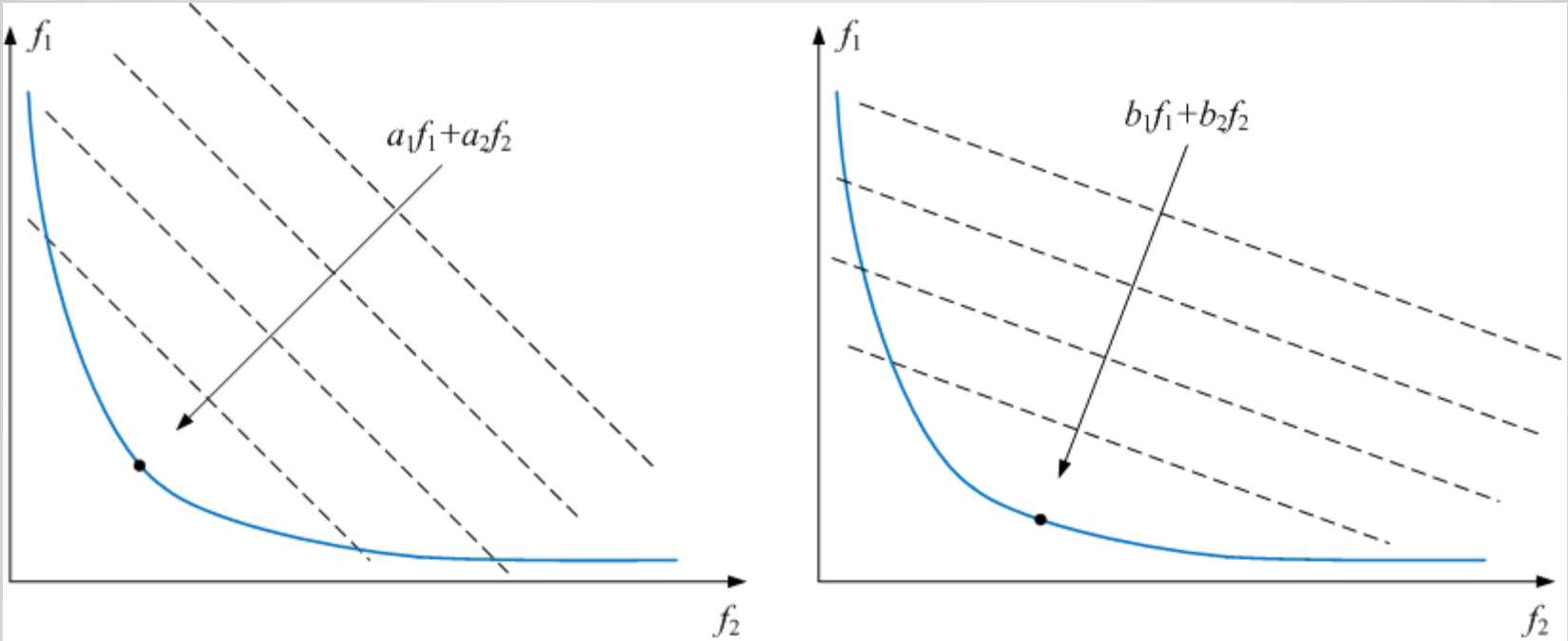
$$\sum_{i=1}^{m} a_i f_i(x) \qquad a_i > 0, \quad i = 1, 2, ..., m$$

Weights a_i may be adjusted depending on the relative importance of particular objective to the designer.

This technique allows finding only a single point on the Pareto optimal solution set.

Solution Techniques for Multiobjective Optimization

Examples of minimizing weighted sum of objectives with different weighting factors:



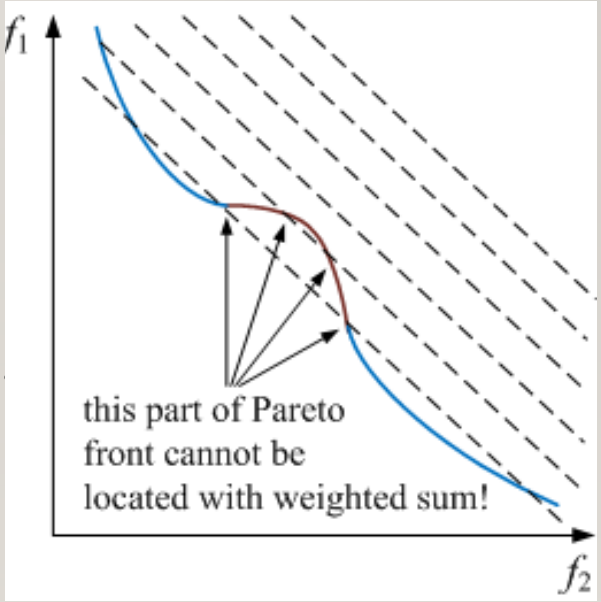
Nowadays, multiple runs with different sets of weights are often performed to generate various points on the Pareto front, which gives the idea of the shape of the front and the information about the trade-off between objectives.

Solution Techniques for Multiobjective Optimization

Drawbacks of the weighted sum approach:

1. It is **difficult to produce a uniform spread of points** on the Pareto front; all the points are often clustered in certain parts of the Pareto optimal set with no points in the interesting “middle” part of the set.

2. **Non-convex parts** of the Pareto front **cannot be obtained** by minimizing convex combinations of the objectives (fortunately non-convex fronts are relatively rare in real-world applications).



Solution Techniques for Multiobjective Optimization

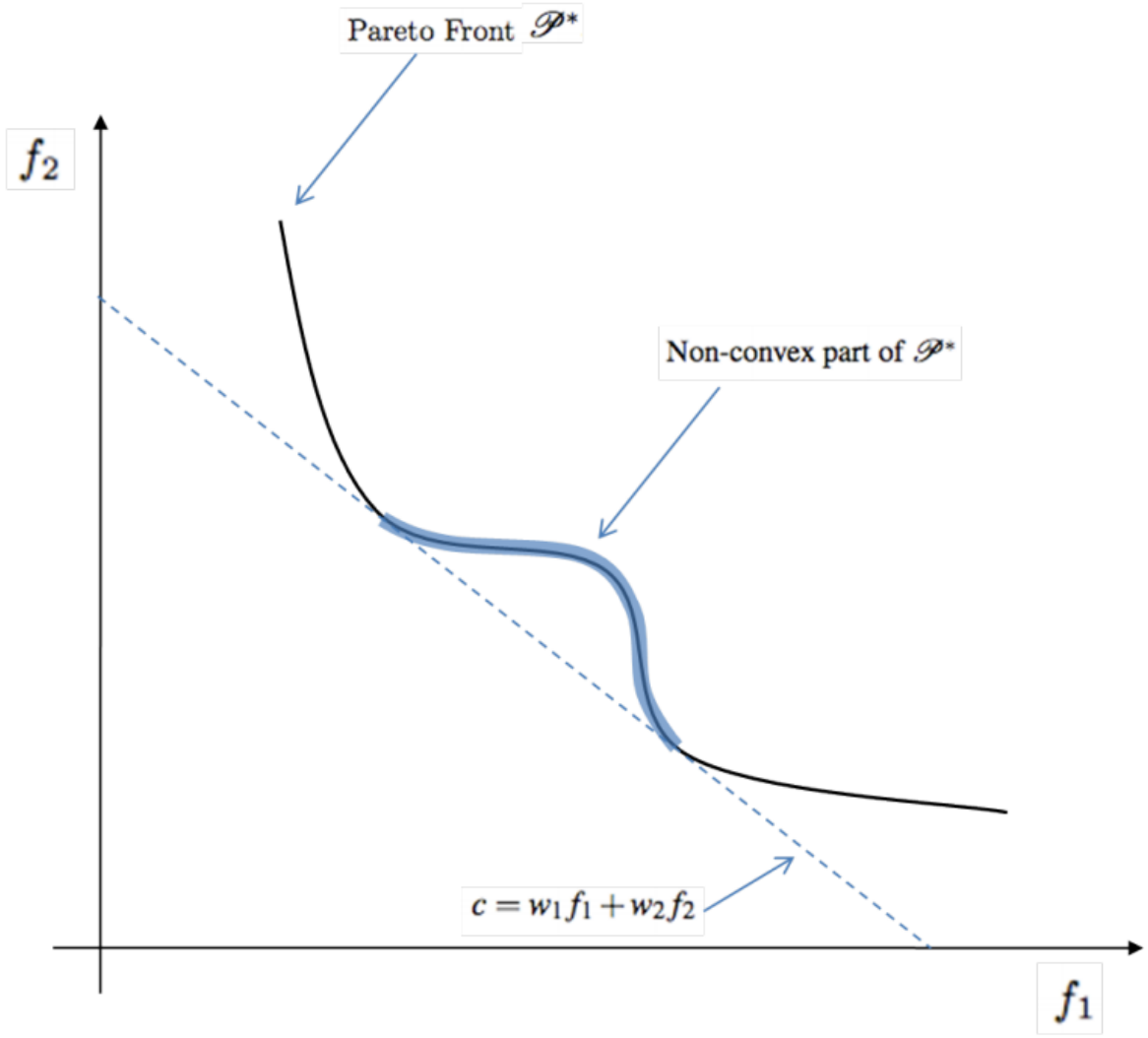


Figure 1: WS is unable to generate the non-convex part of the Pareto front

Solution Techniques for Multiobjective Optimization

Goal attainment method:

Let $F^* = [f_1^* \ \ldots \ f_m^*]^T$ denote a set of design goals associated with a set of objectives $F(x) = [f_1(x) \ \ldots \ f_m(x)]^T$. The relative degree of under- or overachievement of the goals is controlled by a vector of weighting coefficients $w = [w_1 \ \ldots \ w_m]^T$. The optimization problem is formulated as

$$\arg \min_{\gamma \in R, x \in X} \gamma$$

so that

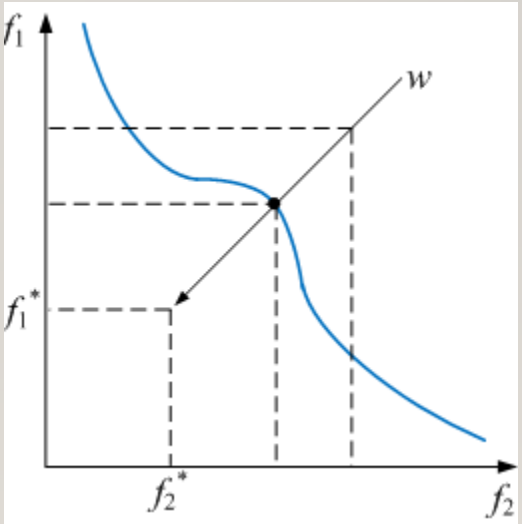
$$f_i(x) - w_i \gamma \leq f_i^* \qquad i = 1, \ldots, m$$

Solution Techniques for Multiobjective Optimization

The term $w_i\gamma$ introduces an element of slackness into the problem, which otherwise imposes that the goals be rigidly met. Weights w_i enable the designer to express a measure of the relative trade-offs between the objectives, e.g., if the weighting vector w equals the initial goals then the same percentage of under- or overattainment of the goals are achieved.

Hard constraints on the goals can be incorporated by setting particular weighting factors w_i to zero.

Goal attainment method allows locating **non-convex** areas of the Pareto front.



Matlab Optimization Toolbox: Goal Attainment Problem

fgoalattain

Solve multiobjective goal attainment problems

Equation

Finds the minimum of a problem specified by

$$\underset{x,\gamma}{\text{minimize}}\gamma \quad \text{such that} \quad \begin{cases} F(x) - weight \cdot \gamma \le goal \\ c(x) \le 0 \\ ceq(x) = 0 \\ A \cdot x \le b \\ Aeq \cdot x = beq \\ lb \le x \le ub. \end{cases}$$

weight, *goal*, *b*, and *beq* are vectors, *A* and *Aeq* are matrices, and $c(x)$, $ceq(x)$, and $F(x)$ are functions that return vectors. $F(x)$, $c(x)$, and $ceq(x)$ can be nonlinear functions.

x, *lb*, and *ub* can be passed as vectors or matrices; see Matrix Arguments.

Syntax

```
x = fgoalattain(fun,x0,goal,weight)
x = fgoalattain(fun,x0,goal,weight,A,b)
x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq)
x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub)
x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon)
x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon,options)
x = fgoalattain(problem)
[x,fval] = fgoalattain(...)
[x,fval,attainfactor] = fgoalattain(...)
[x,fval,attainfactor,exitflag] = fgoalattain(...)
[x,fval,attainfactor,exitflag,output] = fgoalattain(...)
[x,fval,attainfactor,exitflag,output,lambda] = fgoalattain(...)
```

Multiobjective Evolutionary Algorithms

It turns out, that **evolutionary algorithms** are **suitable tools** for finding a good representation of the **Pareto front**.

In particular, it is possible to design an algorithm so that **the population of individuals** is evolving to constitute a set of non-dominated solutions that tend **to move towards a Pareto frontier.**

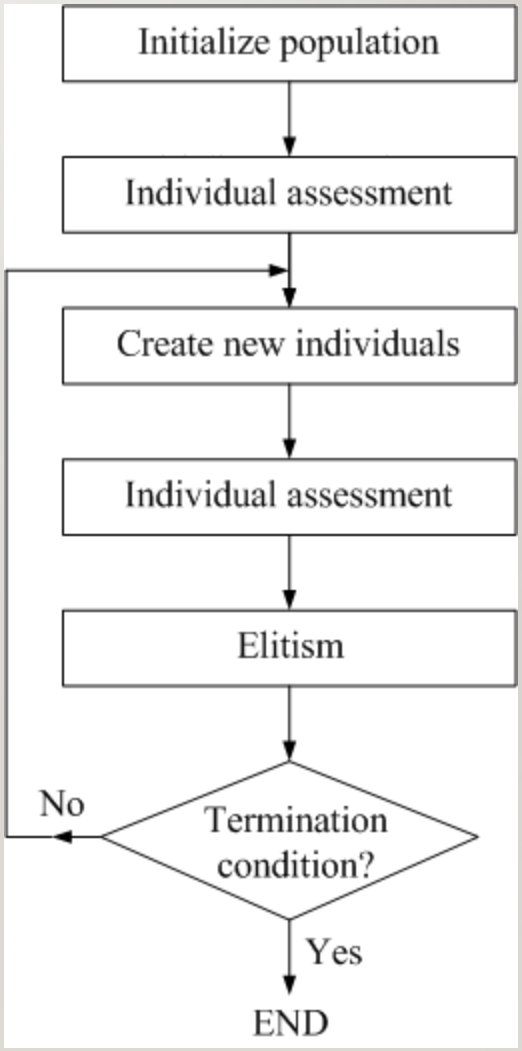
Multiobjective Evolutionary Algorithms

Because of the fact that **evolutionary algorithm** deals with **the set of solutions (population)** anyway, the typical cost of finding the Pareto optimal set is not much higher than a cost of population-based single-objective optimization.

On the contrary, previously discussed methods require a separate single-objective optimization process to find a single point on a Pareto front.

Multiobjective Evolutionary Algorithms: Structure

General structure of
the multiobjective EA



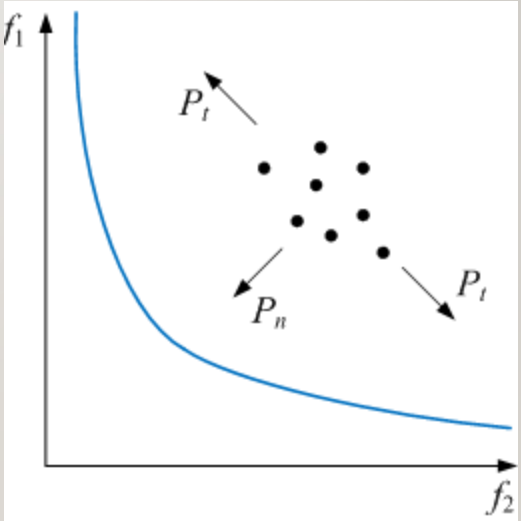
Initialization and creating of the new individuals is **similar** to that of **single-objective EA**

In particular, new individuals are created using recombination and mutation operators (representation and particular operator design depends on the problem)

Multiobjective Evolutionary Algorithms: Structure

Two main factors that differ multi- from single-objective EAs are:

1. Assessment of individuals: members of the population must be assessed so that the preference is given to **nondominated solutions** (normal pressure P_n). At the same time, a **tangent pressure** P_t should be applied in order **to avoid clustering** and to obtain a uniform distribution of solutions along the Pareto front.



2. **Elitism/archiving**: one needs to keep record of a set of the best-found nondominated individuals (elitist individuals) in the evolution in order to achieve a better convergence.

Multiobjective Evolutionary Algorithms: Individual Assessment

Assessment of individuals is based on the **domination relation** $\preceq$

One of possible methods is **ranking**: the rank r_i of an individual i is given as

$$r_i = 1 + q_i$$

where q_i is the number of individuals that dominate individual i.

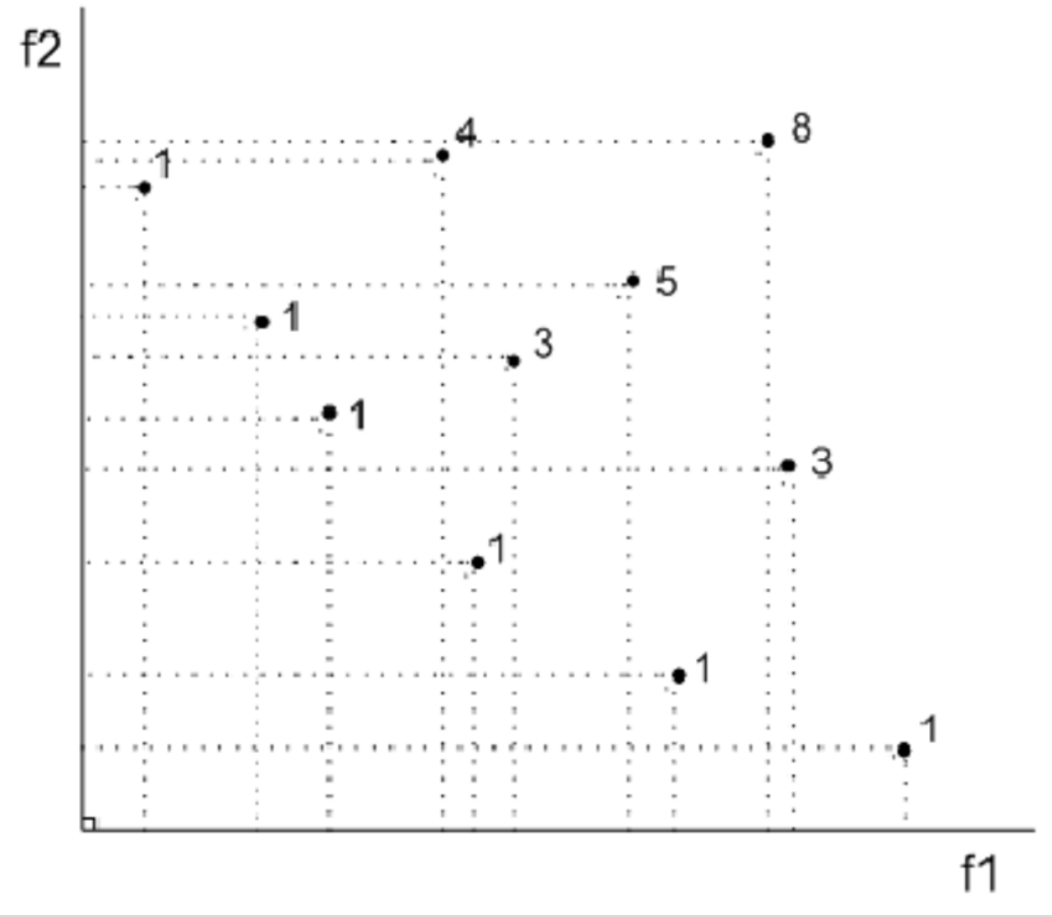
Subsequently, the fitness f_i of an individual i is computed using a mapping inversely related to r_i so that $f_i > f_j$ if and only if $r_i < r_j$. In particular, we can use $f_i = 1/r_i$.

Multiobjective Evolutionary Algorithms: Individual Assessment

One of possible methods is **ranking**: the rank r_i of an individual i is given as

$$r_i = 1 + q_i$$

where q_i is the number of individuals that dominate individual i.



Multiobjective Evolutionary Algorithms: Fitness Sharing

In order to enforce individuals to uniformly spread along the Pareto front, a fitness sharing must be implemented that **penalizes individuals if they are too close to each other.**

Multiobjective Evolutionary Algorithms: Fitness Sharing

Distance between individuals *i* and *j* in the feature space:

$$d\left(F(x^{(i)}), F(x^{(j)})\right) = \left\| F(x^{(i)}) - F(x^{(j)}) \right\|_2$$

Sharing function:

$$SF(i,j) = \begin{cases} 1 - \left[d\left(F(x^{(i)}), F(x^{(j)})\right) / \sigma_{share} \right]^{\alpha} & \text{if } d\left(F(x^{(i)}), F(x^{(j)})\right) < \sigma_{share} \\ 0 & \text{otherwise} \end{cases}$$

(not in the neighborhood)

where α is a parameter that regulates the shape of the sharing function, while σ_{share} determines **the extent of sharing (niche)** allowed in terms of the radius distance

Multiobjective Evolutionary Algorithms: Fitness Sharing

With the sharing information, the shared fitness of any individual *i* is given as

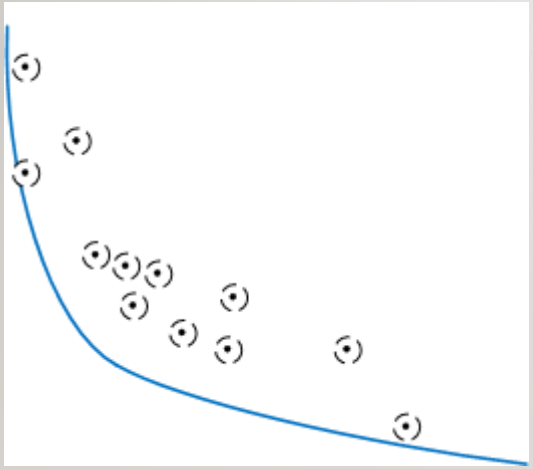
$$f_i' = \frac{f_i}{\sum_{j=1}^{N} SF(i, j)}$$

Thus, individuals in a “crowded neighborhood” have their fitness decreased, which promotes uniform distribution in the population.

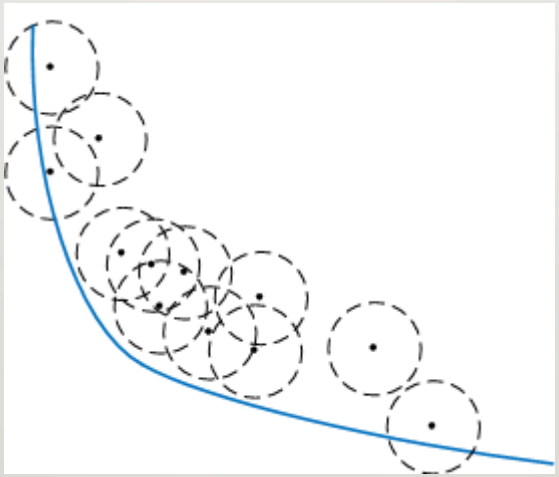
Multiobjective Evolutionary Algorithms: Dynamic Sharing

The sharing distance σ_{share} should be chosen so that the size of the niche is in a proper relation to the number of individuals in the population and the size of the Pareto front.

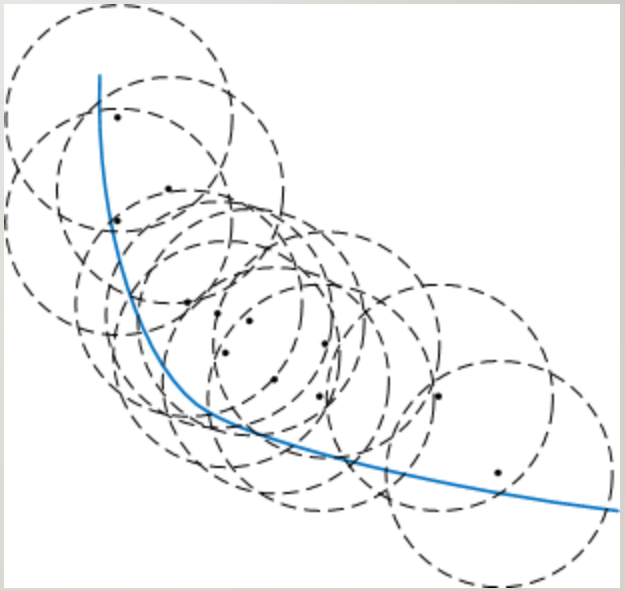
Multiobjective Evolutionary Algorithms: Dynamic Sharing



σ_{share} to small: fitness is not penalized even if individuals are too close to each other



σ_{share} has a right value: fitness is penalized in crowded regions but not elsewhere



σ_{share} to large: fitness is penalized in all regions of the feature space

Too small or too large σ_{share} does not allow to properly detect regions of the feature space which are over-crowded.

Multiobjective Evolutionary Algorithms: Dynamic Sharing

Dynamic sharing allows adjusting σ_{share} to account for the current estimate of the size (volume) of the Pareto front in relation to the number of individuals in the population.

Sharing distance at generation k: $\sigma_{share}^{(k)} = N^{1/(m-1)} \dfrac{d^{(k)}}{2}$

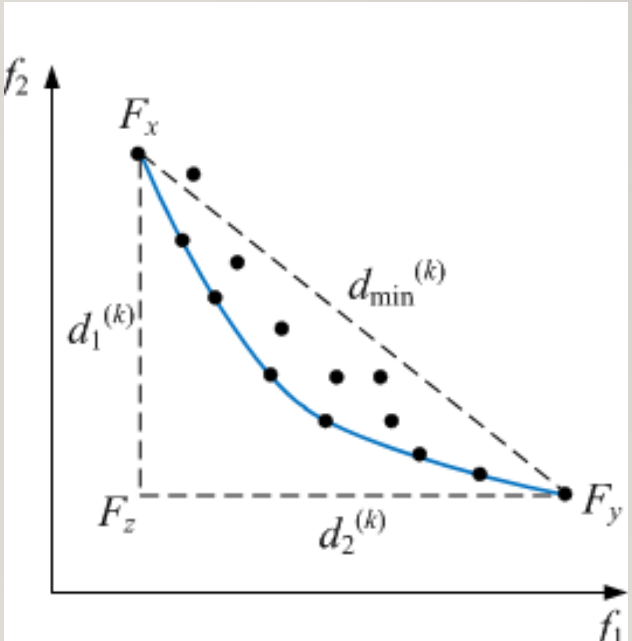
$$d^{(k)} = \left(d_{\max}^{(k)} + d_{\min}^{(k)}\right)/2$$

$$d_{\min}^{(k)} = \sqrt{\left(d_1^{(k)}\right)^2 + \left(d_2^{(k)}\right)^2} \qquad d_{\max}^{(k)} = d_1^{(k)} + d_2^{(k)}$$

$$d_1^{(k)} = \left\| F_x^{(k)} - F_z^{(k)} \right\| \qquad d_2^{(k)} = \left\| F_y^{(k)} - F_z^{(k)} \right\|$$

$$\left\| F_x^{(k)} - F_y^{(k)} \right\| = \max_{i,j \in \{1,\ldots,m\},\, i \neq j} \left\| F(x^{(i)}) - F(x^{(j)}) \right\|$$

$$F_z^{(k)} = \left[f_{z.1}^{(k)} \ \ldots \ f_{z.m}^{(k)} \right]^T \qquad f_{z.j}^{(k)} = \min_{j \in \{1,\ldots,m\}} \left\{ f_{x.j}^{(k)}, f_{y.j}^{(k)} \right\}$$



Multiobjective Evolutionary Algorithms: Selection

Selection of individuals for the next generation in the multiobjective evolutionary algorithm may be based on any selection procedure for single-objective algorithm provided that a **domination-based assessment with fitness sharing** is used to evaluate individuals

Selection may also be based on utilizing the dominance relation in **tournament-like** procedure (**Pareto dominance tournaments**)

Multiobjective Evolutionary Algorithms: Selection

Selection procedure using Pareto dominance tournaments (c_1, c_2: candidates, *comp_set*: comparison set, s: selected individual):

```
c1_dominated = false; c2_dominated = false;
if c1 is dominated by any individual from comp_set,   c1_dominated = true;  end
if c2 is dominated by any individual from comp_set,   c2_dominated = true;  end
if c1_dominated == c2_dominated
    compute shared fitness f1 for individual c1;
    compute shared fitness f2 for individual c2;
    if f1 > f2          %choose a better fit
        s = c1;
    else
        s = c2;
    end
else
    if c1_dominated == false        %choose the non-dominated one
        s = c1;
    else
        s = c2;
    end
end
```

Multiobjective Evolutionary Algorithms: Elitism

multiobjective EA **should implement elitism** in order to keep record of a set of the best non-dominated individuals

The set of best individuals (also called an elitist set) is actually the outcome of the algorithm, as it is a representation of the Pareto front which we were looking for

Also, the elitist set can be optionally used in the reproduction process for faster convergence; this approach should be implemented carefully in order to avoid premature convergence due to over-influence of the best individuals

Multiobjective Evolutionary Algorithms: Elitism

Implementation of elitism (X – set consisting of the current elitist individuals and the newly created population, X' – new elitist set)

Batch mode:
1.Evaluate all individuals in X (i.e., compute shared fitness);
2.Select the subset of best individuals to fill X'.

Recurrence mode:
1.Evaluate all individuals in X;
2.Remove the least promising individuals from X and fill in X';
3.If the size of X' is OK, terminate; else set $X = X'$ and go to 1.

Recurrence mode requires more computational effort, however, it has a higher tendency of avoiding the extinction of local individuals leading to discontinuity of Pareto front

Multiobjective Evolutionary Algorithms: Mating Restrictions

It is important to avoid “production” of poorly fitted individuals (in terms of Pareto dominance) during the execution of the algorithm

So-called *mating restriction* addresses the fact that individuals too different from each other are generally less likely than similar individuals to produce fit offspring through crossover; therefore, mating of similar individuals is favored

Multiobjective Evolutionary Algorithms: Mating Restrictions

Implementation:

1. Specify how close individuals should be in order to mate; set the corresponding parameter σ_{mate}
2. After selection, choose one individual from the mating pool and search for a mate within the distance σ_{mate}
3. If such an individual can be found, the mating is performed; otherwise, a random individual is chosen

Multiobjective Evolutionary Algorithms: Stopping Criteria

Convergence of the multiobjective evolutionary algorithm can be assessed in the sense of progress toward the Pareto front formed by the current nondominated individuals

Practical example of convergence measure:

$$p^{(k)} = \frac{\overline{N}_{nondom}^{(k)}}{N_{nondom}^{(k)}}$$

where $\overline{N}_{nondom}^{(k)}$ is the number of nondominated individuals at generation k that are dominating the nondominated individuals at generation k-1, while $N_{nondom}^{(k)}$ is the number of nondominated individuals at generation k

Multiobjective Evolutionary Algorithms: Stopping Criteria

$p^{(k)}$ approaching zero indicates no further possibility of improvement

In order to avoid irregularity (there might be significant fluctuations of $p^{(k)}$ from iteration to iteration), one can consider an average value of the form:

$$\overline{p}^{(k)} = \sum_{j=k+1-l}^{k} p^{(j)} \Big/ l$$

where $l > 1$

Multiobjective Evolutionary Algorithms: Example

Consider the two-objective optimization problem $f_f(x) = [f_{f1}(x)\ f_{f2}(x)]^T$:

$$f_{f1}(x) = f_{f1}(x_1, ..., x_8) = 1 - \exp\left(-\sum_{j=1}^{8} \left(x_j - \frac{1}{\sqrt{8}} \right)^2 \right)$$

$$f_{f2}(x) = f_{f2}(x_1, ..., x_8) = 1 - \exp\left(-\sum_{j=1}^{8} \left(x_j + \frac{1}{\sqrt{8}} \right)^2 \right)$$

where $-2 \leq x_i \leq 2$, $i = 1,\ldots,8$.

Parameters x_1 to x_8 are to be determined so that f_{f1} and f_{f2} are both minimized.

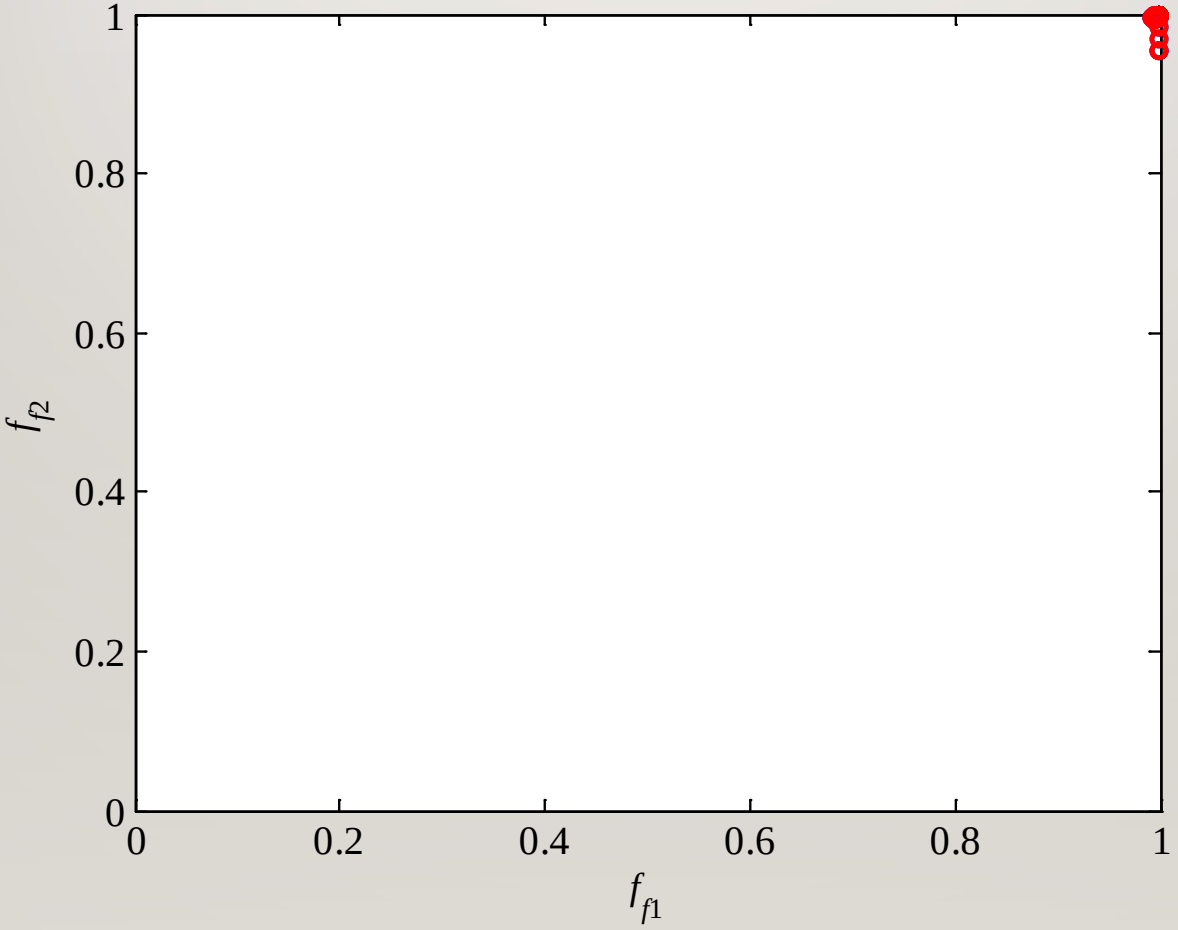
Multiobjective Evolutionary Algorithms: Example

Implementation of evolutionary algorithm:

1. Representation: floating point
2. Crossover: arithmetic, $p_c = 0.7$
3. Mutation: random modification of individual components, $p_m = 0.1$
4. Selection: Pareto dominance tournaments
5. Assessment of individuals: Pareto ranking with dynamic sharing
6. Mating restriction with $\sigma_{mate} = 3 \cdot \sigma_{share}$
7. Population size $N = 100$

Multiobjective Evolutionary Algorithms: Example

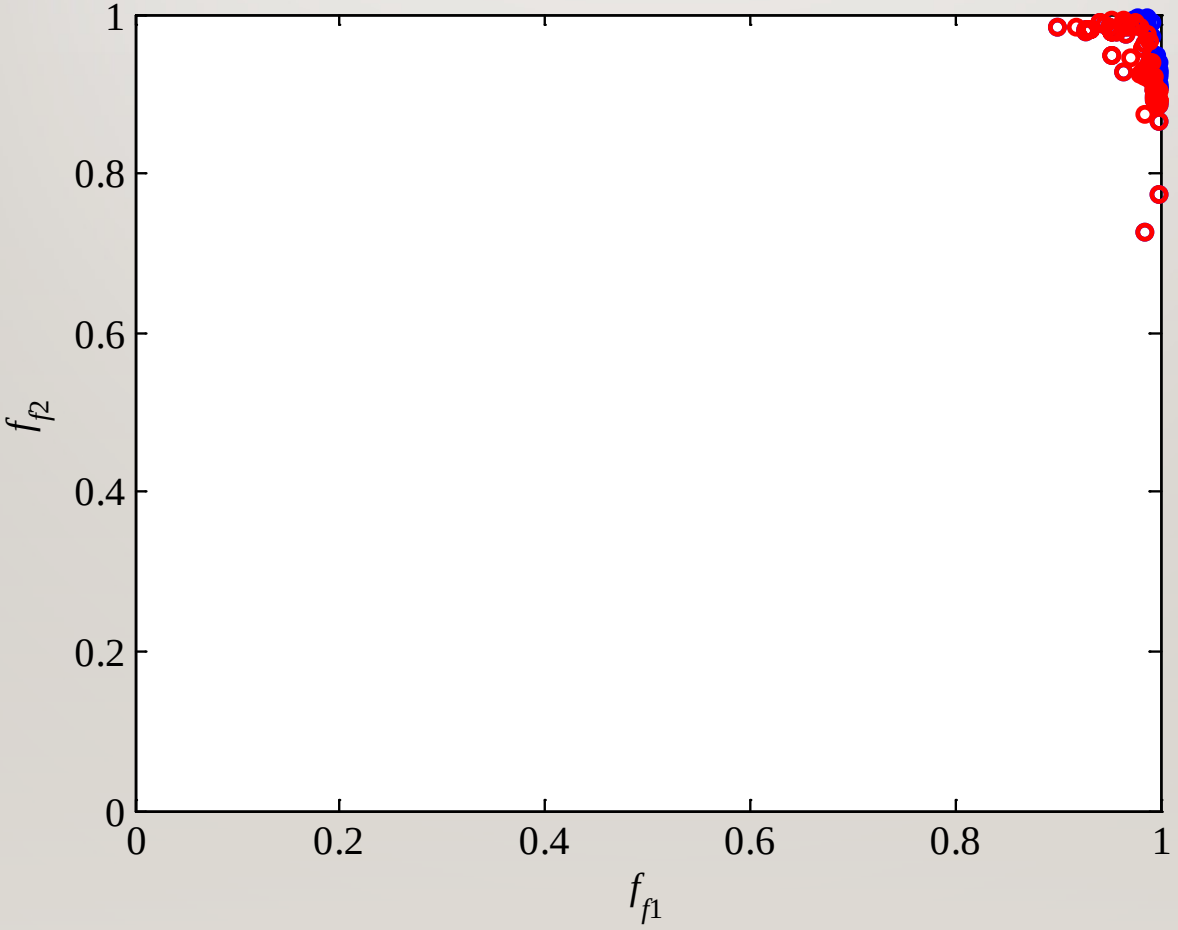
Initial distribution of individuals:



o – current population, o – accumulated elitist individuals

Multiobjective Evolutionary Algorithms: Example

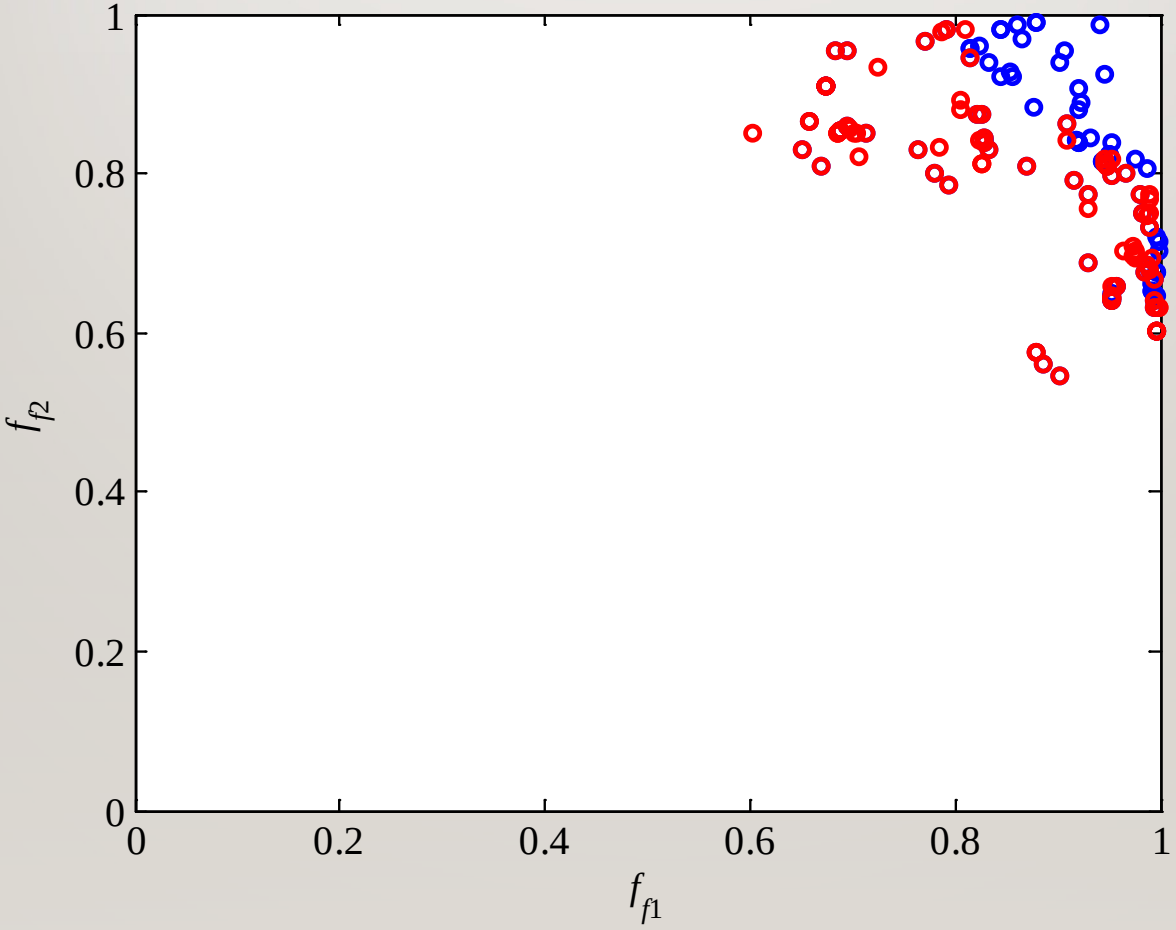
Distribution of individuals after 10 iterations:



o – current population, o – accumulated elitist individuals

Multiobjective Evolutionary Algorithms: Example

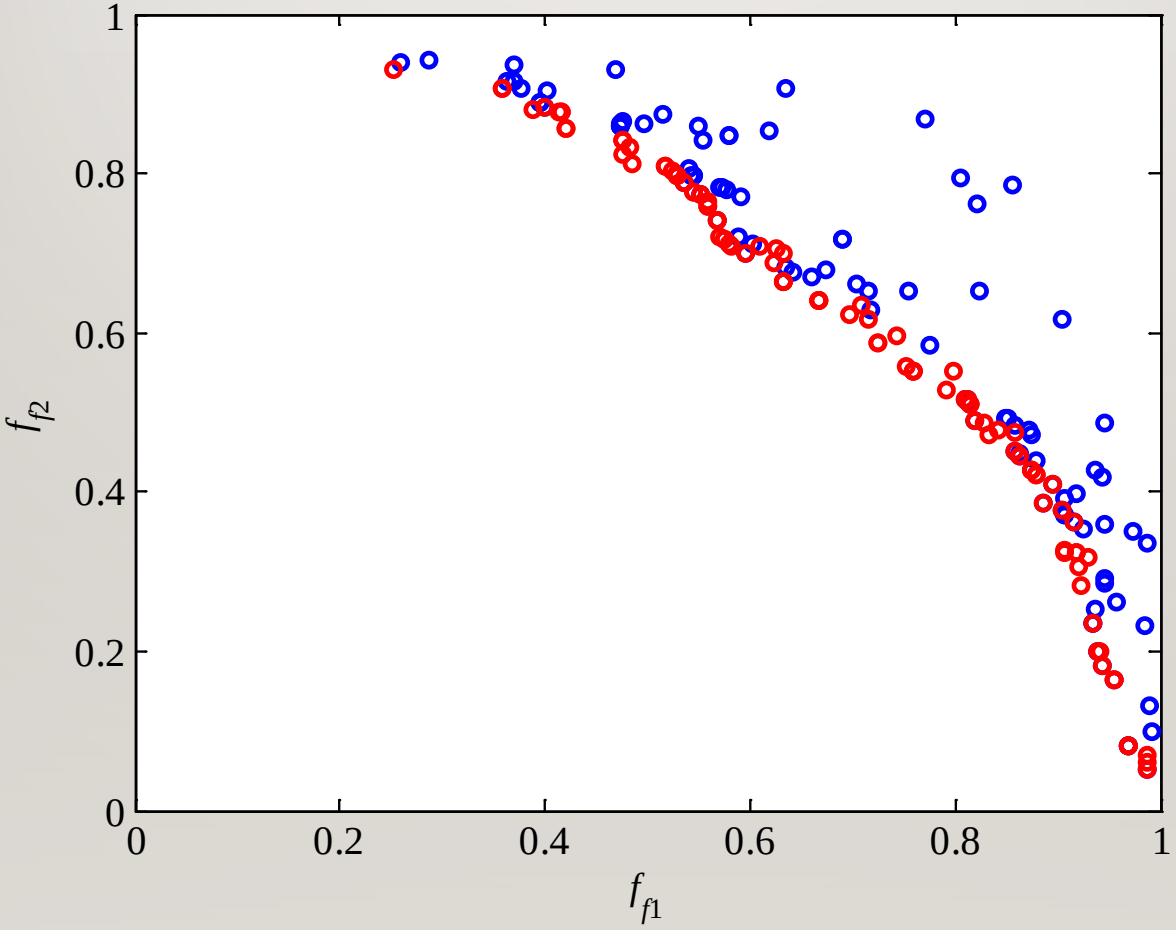
Distribution of individuals after 20 iterations:



o – current population, o – accumulated elitist individuals

Multiobjective Evolutionary Algorithms: Example

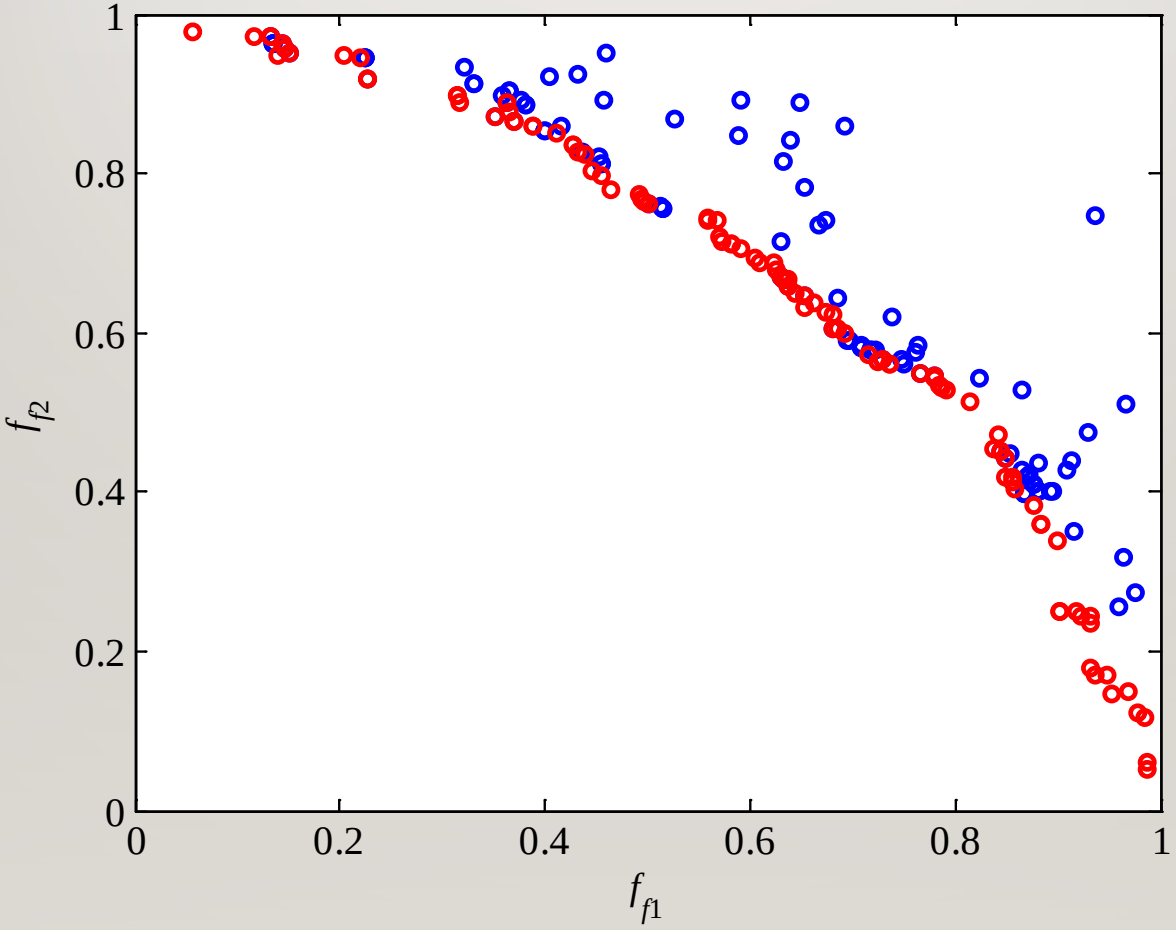
Distribution of individuals after 40 iterations:



o – current population, o – accumulated elitist individuals

Multiobjective Evolutionary Algorithms: Example

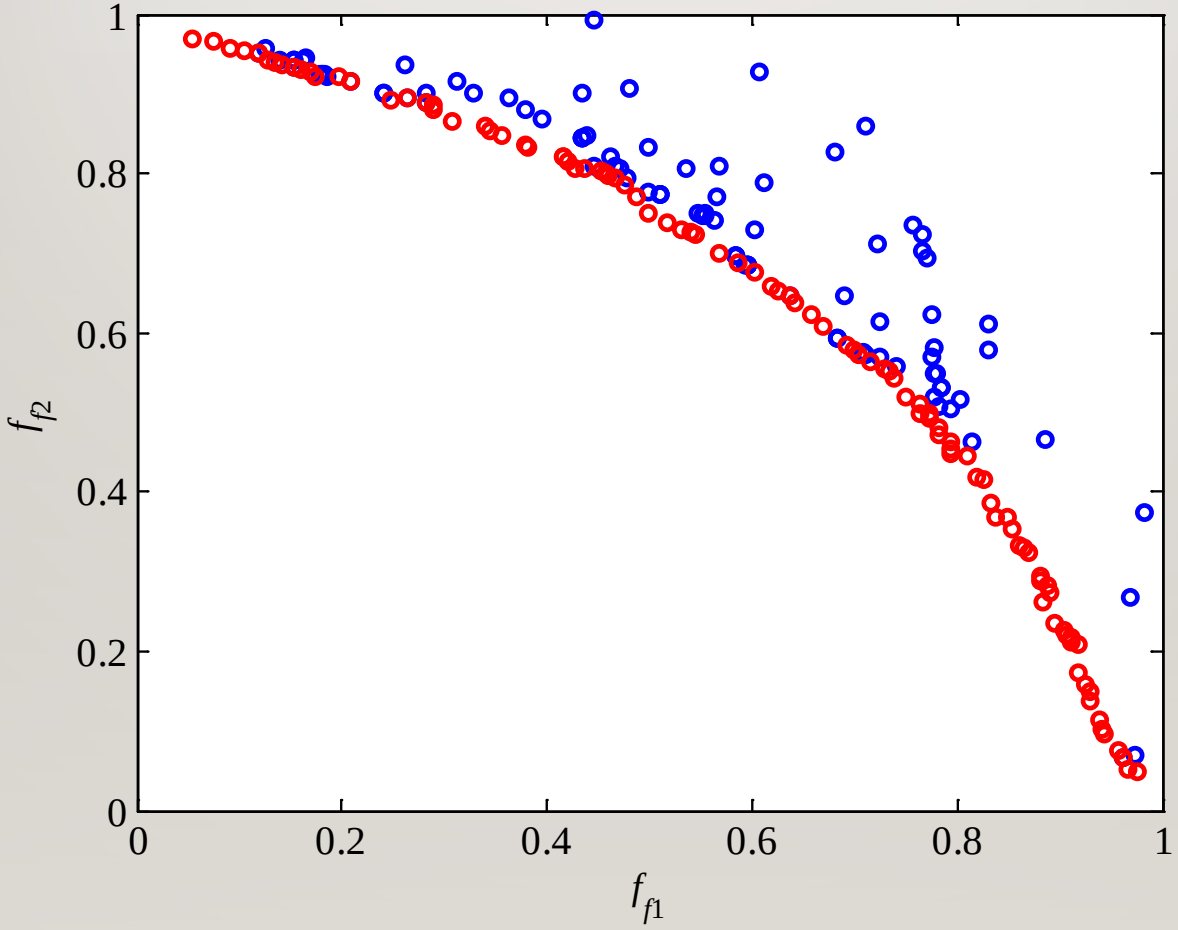
Distribution of individuals after 60 iterations:



o – current population, o – accumulated elitist individuals

Multiobjective Evolutionary Algorithms: Example

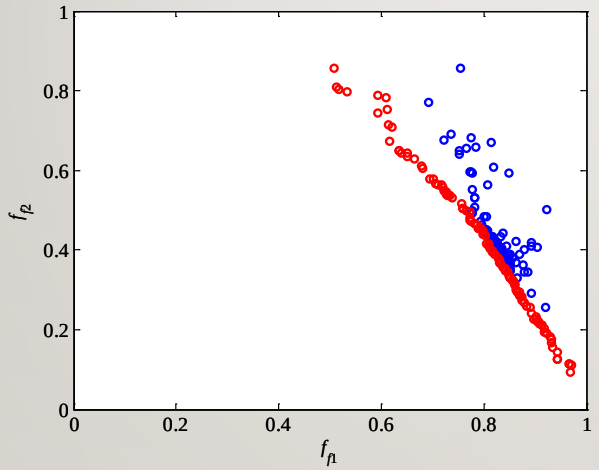
Distribution of individuals upon convergence (179 iterations):



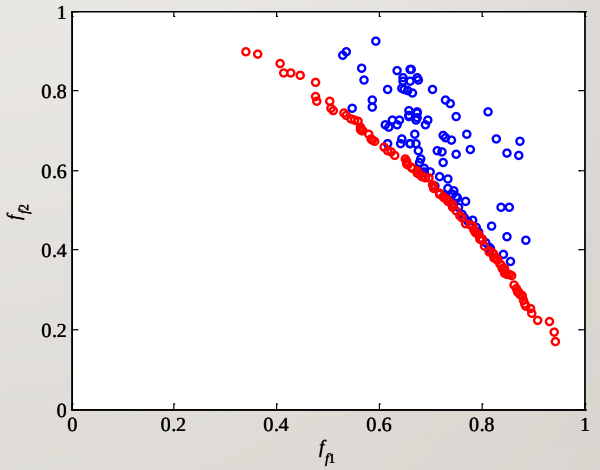
o – current population, o – accumulated elitist individuals

Multiobjective Evolutionary Algorithms: Example

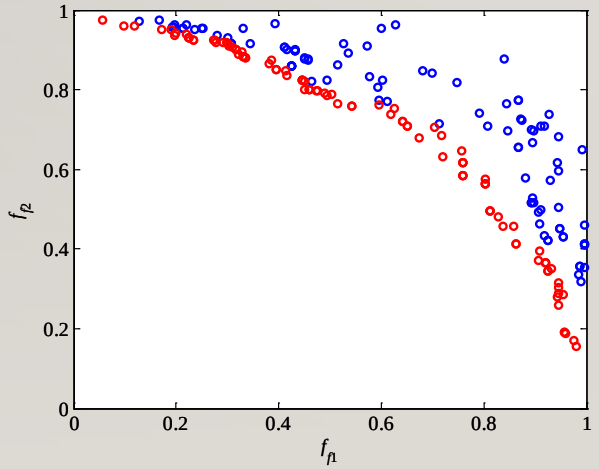
Importance of fitness sharing and mating restrictions:



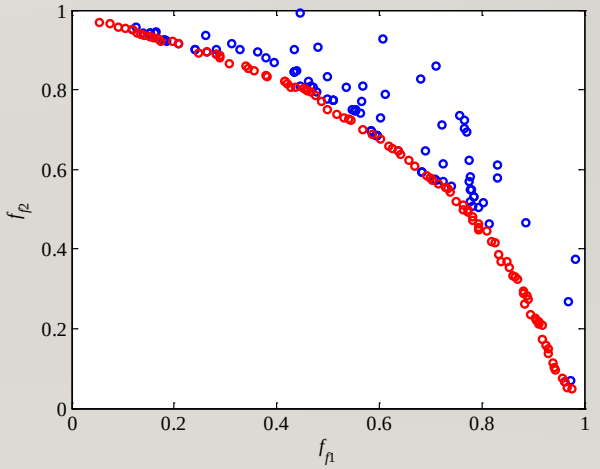
No sharing, no mating restrictions



Sharing but no mating restrictions



No sharing but mating restrictions



Sharing and mating restrictions

Bibliography

D.E. Goldberg, *Genetic algorithms in search, optimization and machine learning*, Addison-Wesley, Reading, MA, 1989.

C.A. Coello Coello, G.B. Lamont, and D.A Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*, 2nd ed, Springer-Verlag, 2007.

K.C. Tan, E.F. Khor, and T.H. Lee, *Multiobjective evolutionary algorithms and applications*, Springer-Verlag, 2005.

C.M. Fonseca, „Multiobjective genetic algorithms with applications to control engineering problems,” PhD thesis, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, UK, 1995.

Test Function f_t

Consider function $f_t = [f_{t1}\ f_{t2}]^T$ defined as:

$$f_{t1}(x) = f_{t1}(x_1, x_2, x_3) = x_1$$

$$f_{t2}(x) = f_{t2}(x_1, x_2, x_3) = \frac{1}{x_1}\left\{1 + \left(x_2^2 + x_3^2\right)^{0.25}\left[\sin^2\left(50\left(x_2^2 + x_3^2\right)^{0.1}\right) + 1\right]\right\}$$

where $0.5 \leq x_1 \leq 1.0$, $-2 \leq x_2, x_3 \leq 2$.

Exercise 1: Multiobjective Using Genetic Algorithm

Test the matlab algorithm gamultiobj using function f_f and f_t.

video tutorial

Exercise 2: Multiobjective Evolutionary Algorithm

Implement a multiobjective evolutionary algorithm using a floating point representation that utilizes the following mechanisms:

1. Selection based on Pareto dominance tournaments
2. Assessment of individuals based on Pareto ranking
3. Dynamic sharing[1]
4. Mating restriction[2]
5. Shared-fitness-based archiving procedure[3]
6. Termination condition based on convergence measure $p^{(k)}$ [4]
7. Visualization procedures to observe current population and elitist individuals

[1] initial implementation may use static sharing with predefined σ_{share}
[2] initial implementation may use regular mating with no restrictions
[3] initial implementation may archive all non-dominated individuals
[4] initial implementation may use maximum number of generations as the termination cond.

Test the algorithm using function f_f and f_t.

Perform experiments in order to tune control parameters, particularly crossover and mutation probability, mutation distribution, mating range, etc.

Perform experiments in order to test the importance of different components of the algorithm, in particular sharing and mating restrictions.